Algorithms for Collaborative and Adversarial Decision Making in Partially Observable Stochastic Games

Shlomo Zilberstein

Resource-Bounded Reasoning Lab

University of Massachusetts, Amherst

Distribution A: Approved for public release; distribution unlimited.

Outline

- What is resource-bounded reasoning
- Computational models of group decision making
- Complexity results and implications
- Algorithms for finite-horizon problems
- Algorithms for infinite-horizon problems
- Conclusion and future directions

Resource-Bounded Reasoning

Problem: Agents cannot be perfectly rational…

- They have limited computational power
- They must react within an acceptable time
- Computation time delays action and reduces the value of the result
- Agents must cope with uncertainty and missing information
- The “appropriate” level of deliberation is situation dependent

Satisficing

- Proposed by Herbert Simon in 1957 to denote decision making that searches until an alternative is found that meets the agent's aspiration level criterion.
- Is satisficing more than a vague principle? Can it be formalized?
- What sort of satisficing models are suitable for computer systems?

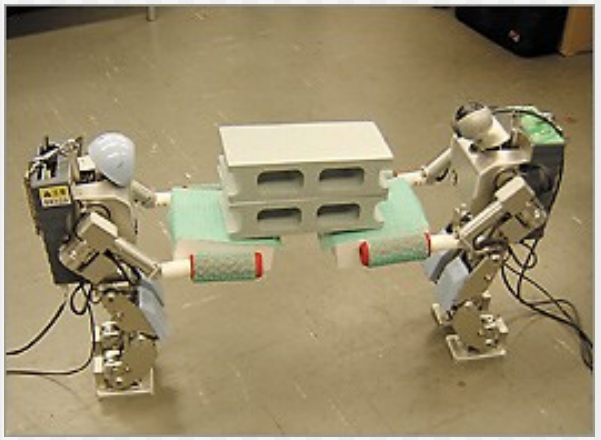
Our Methodology

- Develop computational methods that allow small quantities of resources -- such as time, memory, or information -- to be traded for gains in the value of computed results.
- Develop meta-level control mechanisms that exploit these tradeoffs in order to optimize the overall performance of a system.

Basic Research in RBR

- Developing time- and space-bounded problem-solving algorithms.
- Representation and measurement of computational tradeoffs.
- Embedding resource-bounded reasoning components in large systems.
- Run-time assessment and prediction of solution/decision quality.
- Run-time allocation of computational resources and information gathering.

Decentralized Decision Making

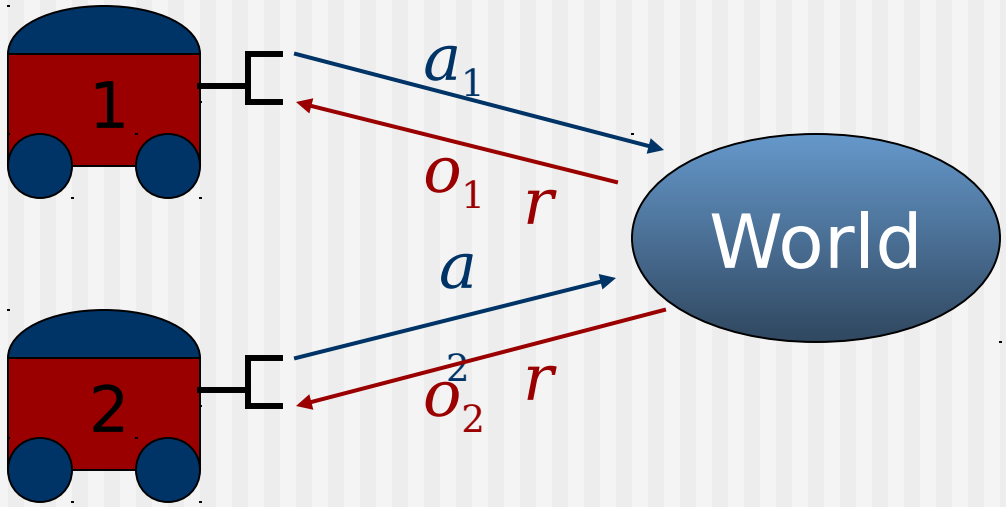


- **Challenge**: How to achieve intelligent coordination of a group of decision makers in spite of stochasticity and limited information?
- Also known as decentralized control, team decision making, multiagent coordination, sequential games

Problem Characteristics

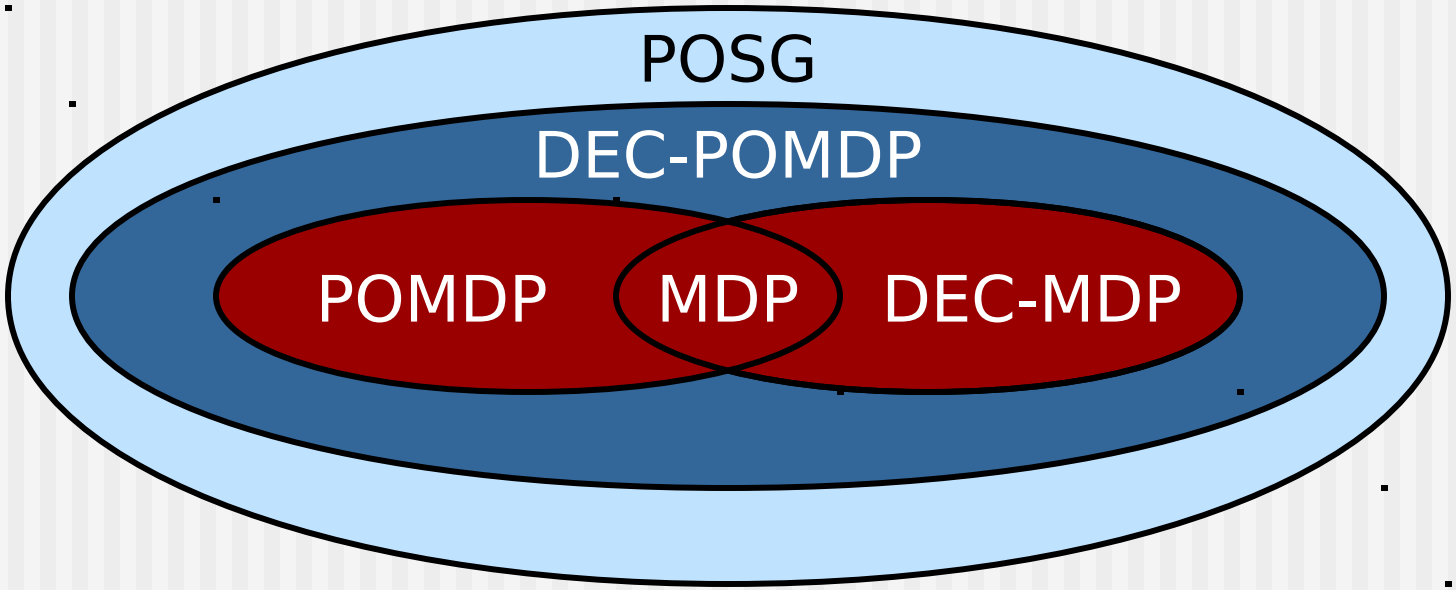
- A **group** of decision makers **interact** with each other in a **stochastic** domain
- Each “episode” involves a **sequence** of decisions over finite or infinite horizon
- State transitions are determined stochastically by the **current state** and the **set of actions**
- Each decision maker has **different partial knowledge** of the global state
- Each decision maker may have **different objectives**

Decentralized POMDP



- Generalization of POMDP involving multiple cooperating decision makers with different observation functions

Formal Models



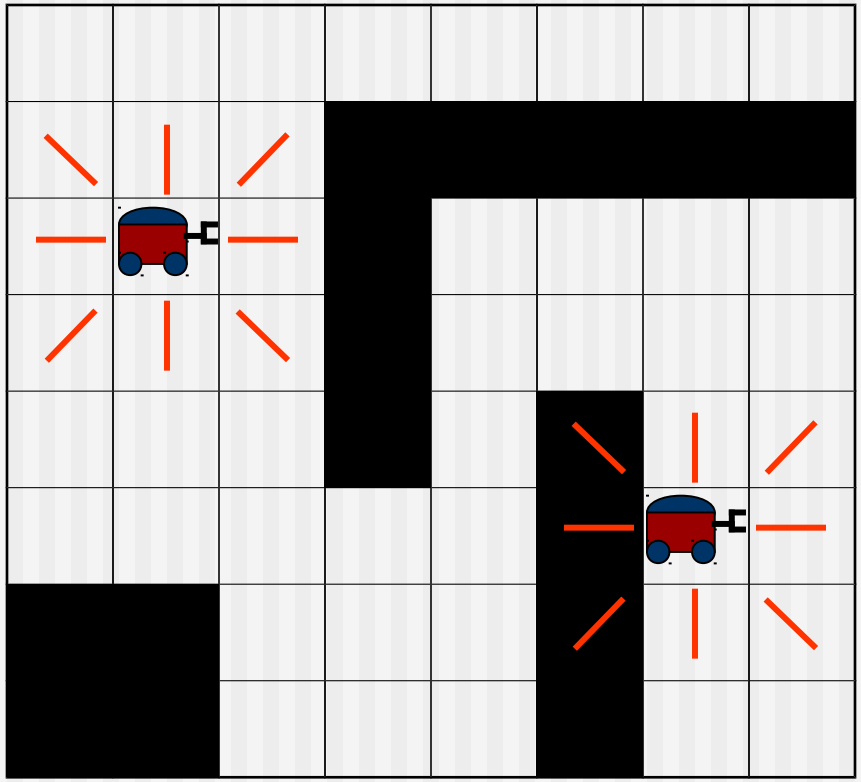
"... there is currently no good way to combine game theoretic and POMDP control strategies."

[Russell & Norvig 03]

DEC-POMDPs & POSGs

- A POSG is ‹S, A_1, A_2, P, R_1, R_2, Ω_1, Ω_2, O›, where
  - S is a finite set of domain states, with initial state s_0
  - A_1, A_2 are finite action sets
  - $P(s, a_1, a_2, s')$ is a state transition function
  - $R_1(s, a_1, a_2)$ and $R_2(s, a_1, a_2)$ are reward functions
  - Ω_1, Ω_2 are finite observation sets
  - $O(a_1, a_2, s', o_1, o_2)$ is an observation function
- A DEC-POMDP is a POSG with a single reward function $R(s, a_1, a_2)$
- Straightforward generalization to n agents

Example: Mobile Robot Planning



States: grid cell pairs

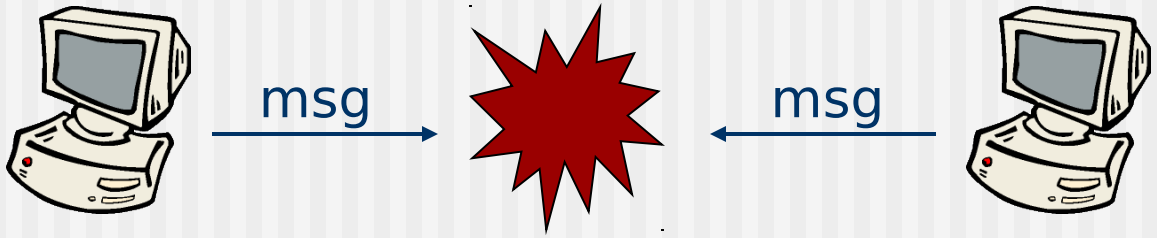
Actions: ↑,↓,←,→

Transitions: noisy

Goal: meet quickly

Observations: red lines

Example: Broadcast Channel



- Messages arrive at nodes randomly
- Collision if two send at once
- Turn-taking strategy not necessarily optimal

States: who has a message to send?
Actions: send or don't send
Reward: +1 for successful broadcast, otherwise 0
Observations: was there a collision?

Solving DEC-POMDPs

- A local policy δ_i is a strategy for agent i
- Can be represented as a mapping from
  - Local **observation sequences** to **actions**; or
  - Local **memory states** to **actions**
- A joint policy is a pair $\langle \delta_1, \delta_2 \rangle$
- Goal is to maximize expected reward over a finite horizon or discounted infinite horizon
- We consider in this talk the problem of centralized, off-line planning and decentralized execution.

Some Fundamental Questions

- Are DEC-POMDPs significantly harder to solve than POMDPs? Why?
- What features of the problem domain affect the complexity and how?
- Is dynamic programming possible?
- Can dynamic programming be made practical?
- Is it beneficial to treat communication as a separate type of action?
- What kind of approximation schemes can be developed?

The Complexity of DEC-POMDPs

Bernstein, Givan, Immerman & Zilberstein, UAI 2000, MOR 2002

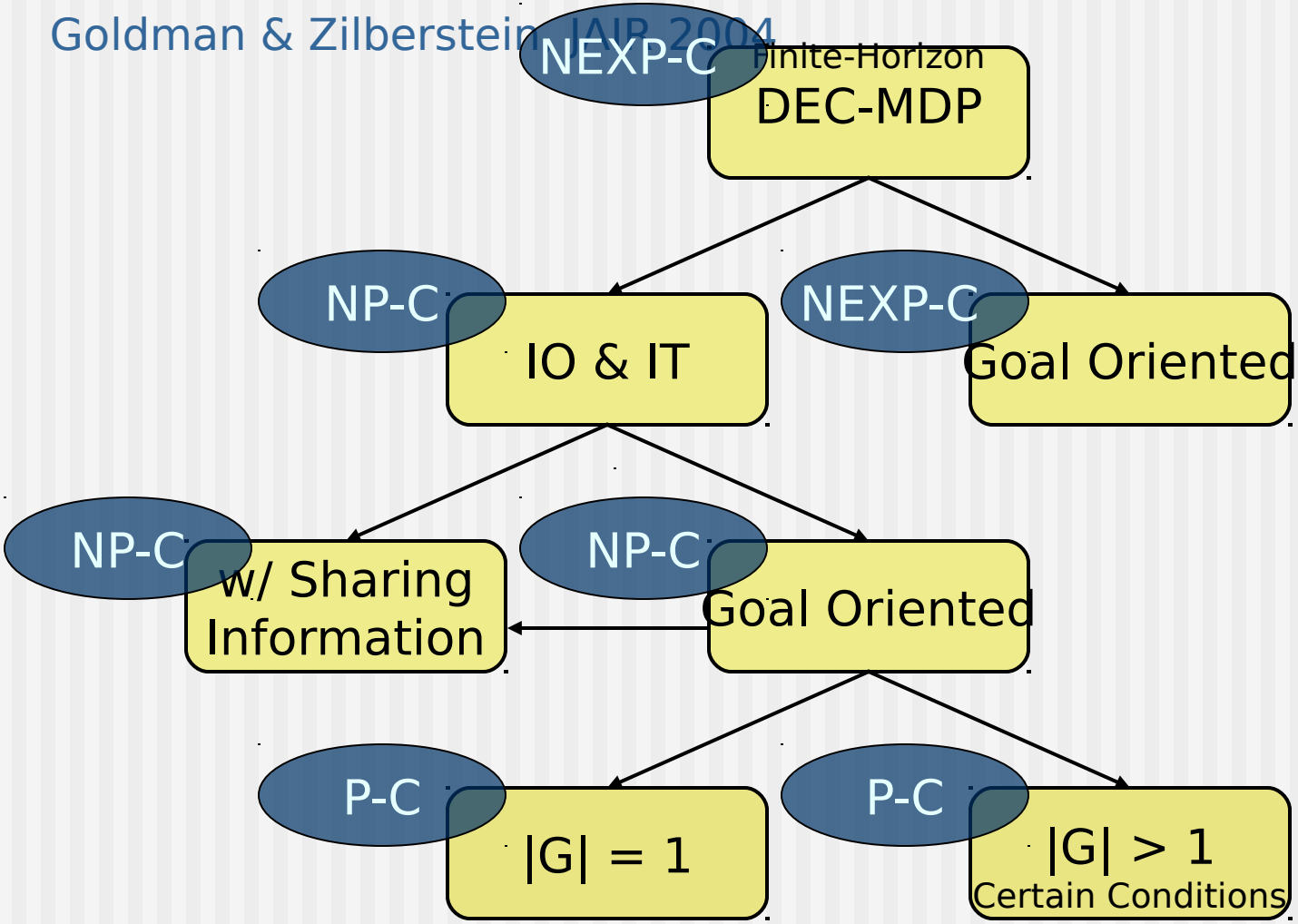
- The complexity of finite-horizon DEC-POMDPs has been hard to establish. In 1985, Tsitsiklis and Athans proved that the **static** version of the problem, where a **single** set of decisions is made in response to a **single** set of observations, is NP-hard [Tsitsiklis and Athan 85]
- We have recently proved that *two-agent finite-horizon DEC-(PO)MDPs are NEXP-hard*
- Hardness comes from **decentralized operation**, not from having hidden state

Special Classes of DEC-POMDPs

- Factored state spaces (structured domains)
- Independent transitions (IT)
- Independent observations (IO)
- Structured reward function (SR)
- Goal-oriented objectives (GO)
- Degree of observability (partial, full, jointly full)
- Degree of information sharing

Complexity of Sub-Classes

Goldman & Zilberstein, JAIR 2004

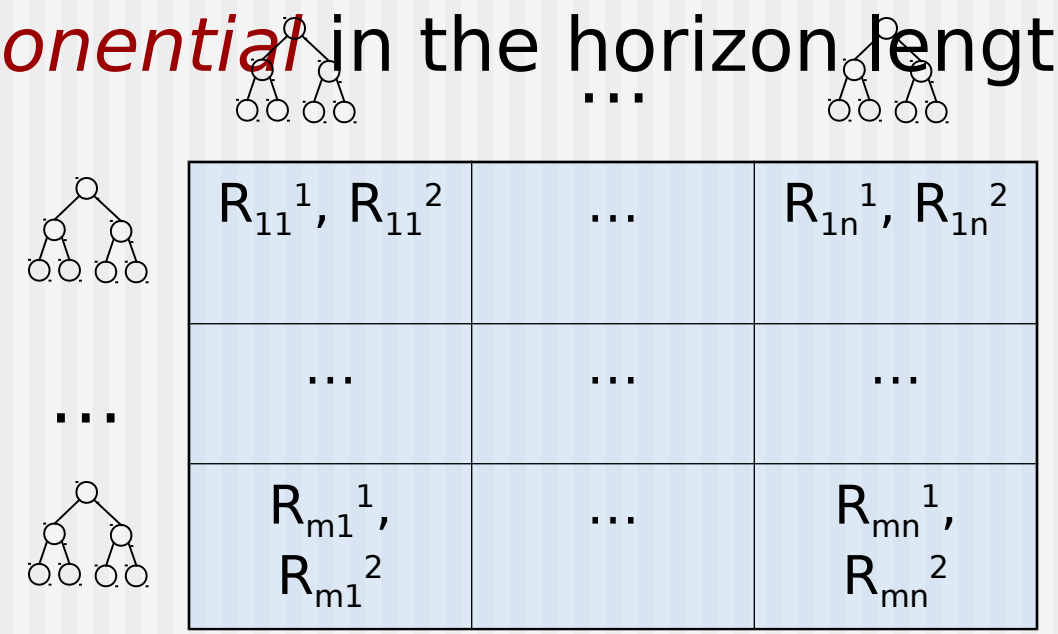


DP for Finite-Horizon DEC-POMDPs

- POMDPs can be viewed as belief-state MDPs
- Value iteration algorithms have been developed [Smallwood & Sondik 73]
- Not clear how to define a belief-state MDP for a DEC-POMDP
- But they can be solved using dynamic programming (DP)

Strategy Elimination

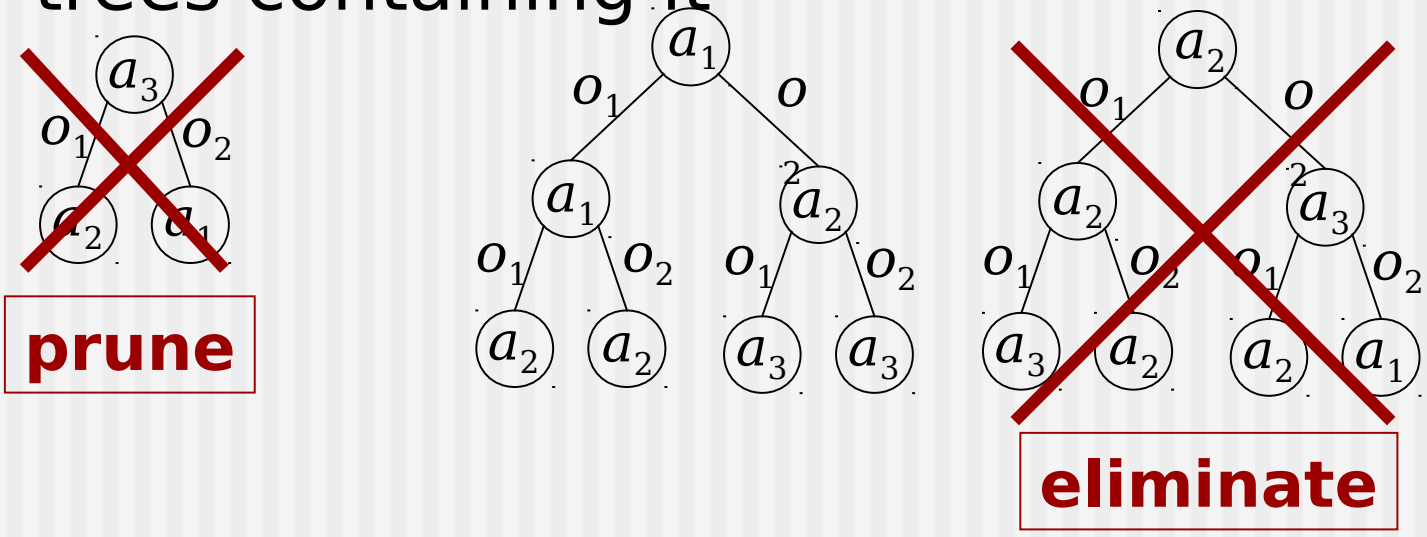
- Could simply convert to normal form game
- But the number of strategies is *doubly exponential* in the horizon length!



A Better Way to Do Elimination

Hansen, Bernstein & Zilberstein, AAAI 2004

- We use dynamic programming to eliminate dominated strategies without first converting to normal form
- Pruning a subtree eliminates the set of trees containing it



Alternative: Heuristic Search

Szer, Charpillet & Zilberstein, UAI 2005

- Take advantage of a known start state distribution & domain-independent heuristics
- The MAA* algorithm:

Initialize open list D with all possible horizon-1 joint policies

Repeat:

1. Select $\delta \in D$ with highest evaluation Value($\delta|s_0$)
2. Expand δ one level and add children to D
3. Remove δ from D

Until the selected δ in step 1 is of horizon T

► When the algorithm stops, δ is optimal

Infinite-Horizon DEC-POMDPs

Bernstein, Hansen & Zilberstein, IJCAI 2005

- Because it is not clear how to define a belief-state for a DEC-POMDP, value iteration does not generalize to the infinite-horizon case
- Can generalize policy iteration for POMDPs [Hansen 98, Poupart & Boutilier 04]
- Basic idea: Representing local policies using stochastic finite-state controllers and define a set of controller transformations that guarantee convergence

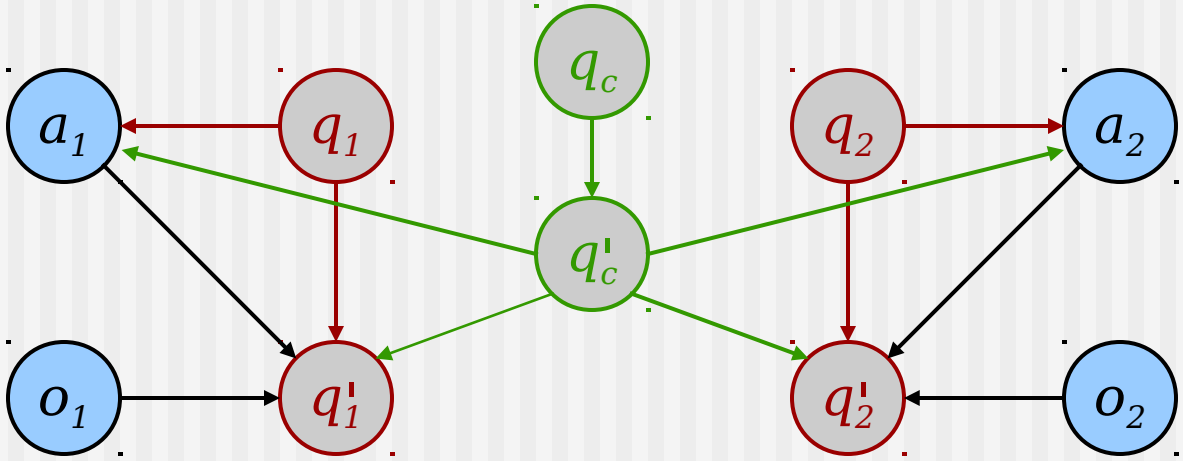
Independent Joint Controllers

- Can define a local controller for agent i to be a conditional distribution $P(a_i, q_i \mid q_i, o_i)$
- An independent joint controller is

  $\Pi_i P(a_i, q_i \mid q_i, o_i)$

Correlated Joint Controllers

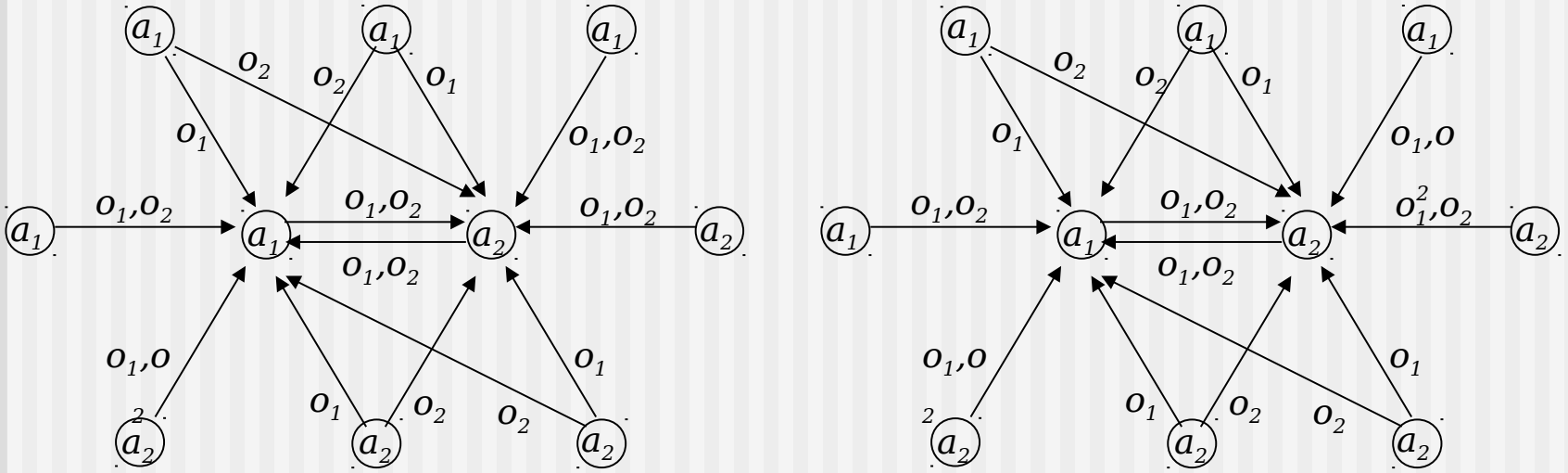
- A correlation device, $[Q_c, \psi]$, is a set of nodes and a stochastic state transition function
- Joint controller is $\sum_{q_c} P(q_c' | q_c) \prod_i P(a_i, q_i' | q_i, o_i, q_c')$



- Random bits for the correlation device can be determined prior to execution time

Exhaustive Backups

- Add a node for every possible action and deterministic transition rule



- Repeated backups converge to optimality, but lead to very large controllers

Value-Preserving Transformations

- A value-preserving transformation changes the joint controller without sacrificing value
- Formally, there must exist mappings

  $f_i : Q_i \to \Delta R_i$ for each agent i and $f_c : Q_c \to \Delta R_c$ such that

$$V(s, q', q_c) \leq \sum_{r} P(r \mid q') \sum_{r_c} P(r_c \mid q_c) V(s, r, r_c)$$

  for all $s \in S$, $q' \in Q'$, and $q_c \in Q_c$

Policy Iteration Algorithm

Bernstein, Hansen & Zilberstein, IJCAI 2005

Repeat

1) Evaluate the controller
2) Perform an exhaustive backup
3) Perform value-preserving transformations

Until controller is ε-optimal for all states

Theorem: *For any ε, policy iteration returns a correlated joint controller that is ε-optimal for all initial states in a finite number of iterations.*

Efficient Transformations

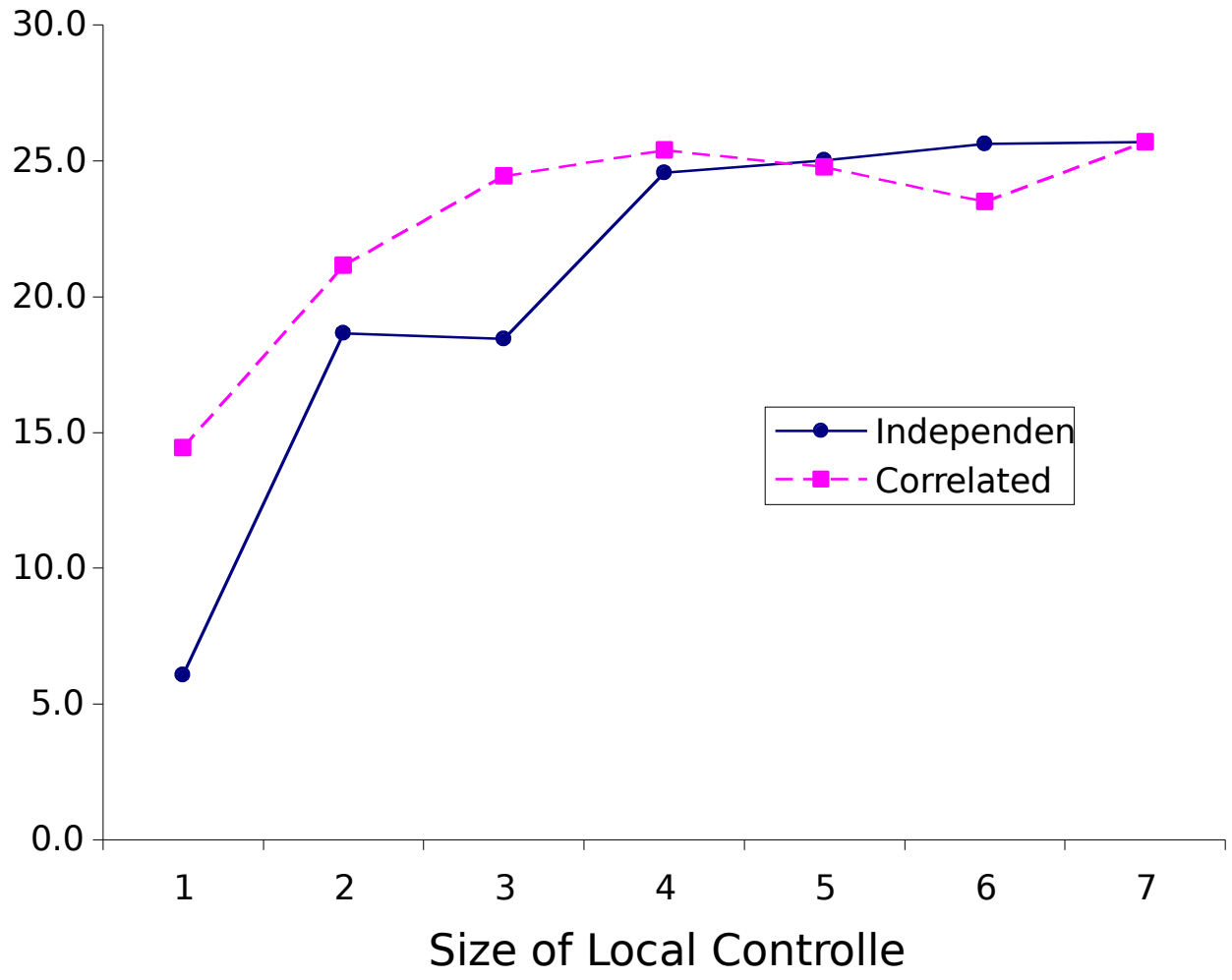
- Controller reductions
  - Shrink the controller without sacrificing value
- Bounded dynamic programming updates
  - Increase value while keeping the size fixed

- Both can be done using polynomial-sized linear programs
- Generalize ideas from POMDP literature [Poupart & Boutilier 03]

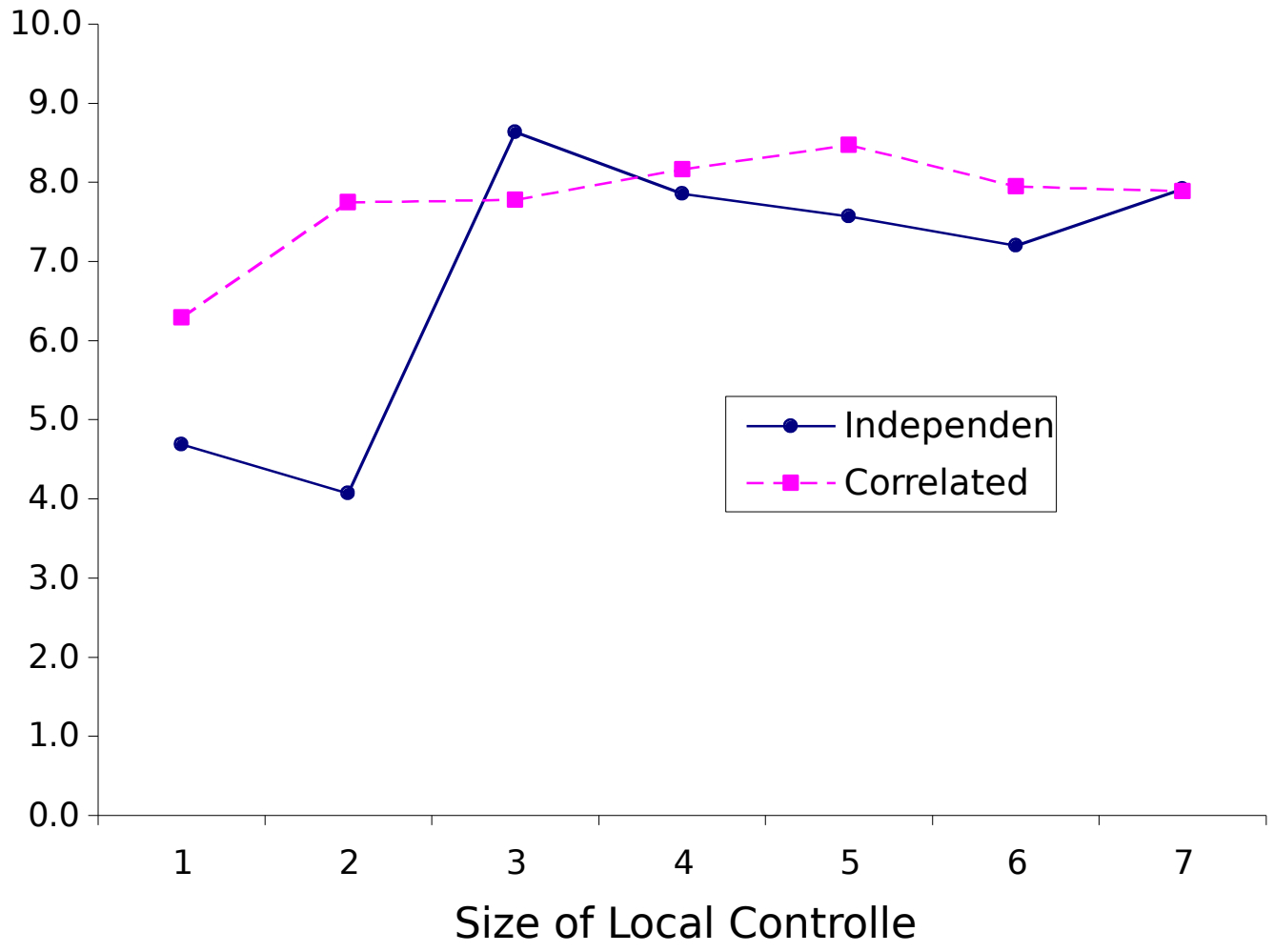
Modifying the Correlation Device

- Transformations can be applied to the correlation device
- Slightly different linear programs to solve
- Can think of the correlation device as another agent
- Lots of implementation questions remain
  - What to use for an initial joint controller?
  - Which transformations to perform?
  - Order for choosing nodes to remove or improve?

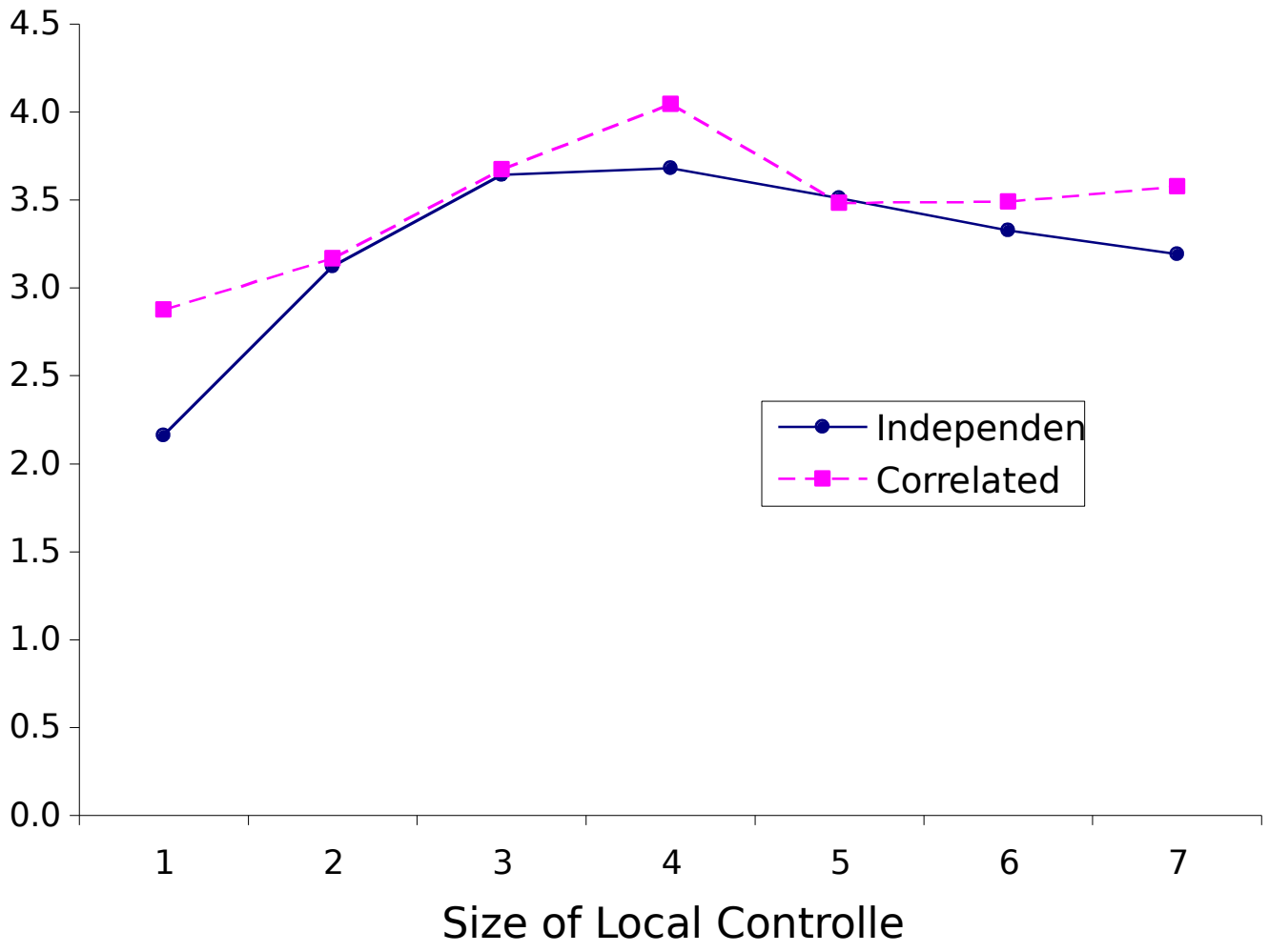
Recycling Robots



Broadcast Channel



Meeting on a Grid



Performance

- For each domain, we found better solutions than with exhaustive backups
- A larger correlation device tends to lead to higher values on average
- Larger local controllers tend to yield higher average values *up to a point*
- Bounded DP is limited by improving one controller at a time

Recap Objectives

- Lay the foundation for developing new algorithms for group decision making
- Improve the understanding of the interaction between complexity and problem structure
- Develop effective new algorithms for solving certain multi-agent problems optimally or approximately
- Expand the scalability of existing solution techniques and address their high computational complexity

Lessons Learned

- There are some fundamental differences between centralized and decentralized operation that make planning much more complicated
- Some DP techniques carry over from centralized (PO)MDPs to the decentralized/adversarial cases, but most need to be modified significantly
- Scalability of algorithms remains a major challenge; exploiting the inherent structure in real-world problems helps cope with complexity
- Realistic models of opponents are unattainable; instead, it is beneficial to model simple tendencies such as well-defined weaknesses

New Directions

http://anytime.cs.umass.edu

- Separate policies for action & communication
- Communication-based decomposition
- Large numbers of weakly-interacting agents
- Extending algorithms to adversarial situations
- Decentralized planning and execution
- Solving the problem through online learning

- Language learning

Questions?

Answer Some Basic Questions

- Are DEC-POMDPs significantly harder to solve than POMDPs?
- What problem domain characteristics affect the complexity and how?
- Is optimal dynamic programming possible?
- Can dynamic programming be made practical?
- Is it beneficial to treat communication as a separate type of action?
- What kind of approximation schemes can be developed?